

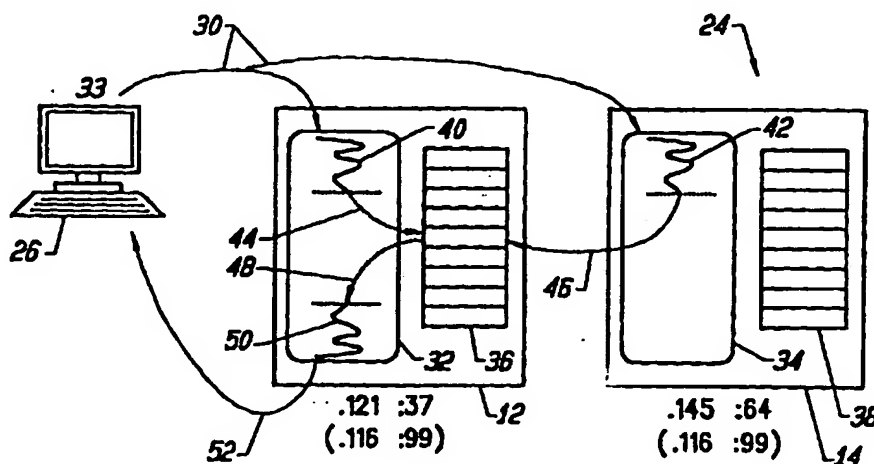
10/639603 190-892

**PCT**WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G06F 11/00		A1	(11) International Publication Number: WO 96/23259
			(43) International Publication Date: 1 August 1996 (01.08.96)
(21) International Application Number: PCT/US96/01007		(81) Designated States: AU, BR, CA, JP, MX, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).	
(22) International Filing Date: 23 January 1996 (23.01.96)			
(30) Priority Data: 08/378,966 27 January 1995 (27.01.95) US		Published With international search report.	
(71) Applicant: AUSPEX SYSTEMS, INC. [US/US]; 5200 Great America Parkway, Santa Clara, CA 95054 (US).			
(72) Inventors: KANDASAMY, David, R.; 199 Victory Circle, San Ramon, CA 94583 (US). BUTLER, Mitchel, B.; 522 North Cascade Terrace, Sunnyvale, CA 94087 (US). FOSS, Andrew, L.; 3457 92nd Avenue, N.E., Yarrow Point, WA 98004 (US). PETERSON, Bradley, M.; 372 Sunkist Lane, Los Altos, CA 94022 (US). PATWARDHAN, Chintamani, M.; 428 Madera Avenue #14, Sunnyvale, CA 94086 (US). RIBBLE, Michael, T.; 484 Wraight Avenue, Los Gatos, CA 95030 (US). ROTHMEIER, Dieter; 90 Hawthorne Way, San Jose, CA 95110 (US). RAMIL, Gaudencio; 1450 Stemel Way, Milpitas, CA 95035 (US).			
(74) Agent: ROSENBERG, Gerald, B.; Fliesler, Dubb, Meyer and Lovejoy, Suite 400, Four Embarcadero Center, San Francisco, CA 94111-4156 (US).			

(54) Title: FAULT TOLERANT NFS SERVER SYSTEM AND MIRRORING PROTOCOL



(57) Abstract

A network computer system providing fault tolerant storage and retrieval of data files including a client system (26) connected to a data communication network that may source a first data transfer request to said data communication network for the transfer or retrieval of data. A first server system (12) and a second server system (14), both including media for storage of data files, are connected to the data communication network. Control protocol, established between the first and second server systems, coordinates an asymmetric response by the first and second server systems to a first data transfer request, such that file data transferred by the client is replicated to the first and second storage media and such that file data transferred to the client system is non-replicatively provided to the client system by either the first or second server system.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	GB	United Kingdom	MW	Malawi
AT	Austria	GE	Georgia	MX	Mexico
AU	Australia	GN	Guinea	NE	Niger
BB	Barbados	GR	Greece	NL	Netherlands
BE	Belgium	HU	Hungary	NO	Norway
BF	Burkina Faso	IE	Ireland	NZ	New Zealand
BG	Bulgaria	IT	Italy	PL	Poland
BJ	Benin	JP	Japan	PT	Portugal
BR	Brazil	KE	Kenya	RO	Romania
BY	Belarus	KG	Kyrgyzstan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic	KR	Republic of Korea	SE	Sweden
CG	Congo	KZ	Kazakhstan	SG	Singapore
CH	Switzerland	LI	Liechtenstein	SI	Slovenia
CI	Côte d'Ivoire	LK	Sri Lanka	SK	Slovakia
CM	Cameroon	LR	Liberia	SN	Senegal
CN	China	LT	Lithuania	SZ	Swaziland
CS	Czechoslovakia	LU	Luxembourg	TD	Chad
CZ	Czech Republic	LV	Latvia	TG	Togo
DE	Germany	MC	Monaco	TJ	Tajikistan
DK	Denmark	MD	Republic of Moldova	TT	Trinidad and Tobago
EE	Estonia	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	UG	Uganda
FI	Finland	MN	Mongolia	US	United States of America
FR	France	MR	Mauritania	UZ	Uzbekistan
GA	Gabon			VN	Viet Nam

-1-

Fault Tolerant NFS Server
System and Mirroring Protocol

5 Background of the Invention

1. Field of the Invention:

 The present invention is generally related to
 fault tolerant computer systems and, in particular, to
 computer systems that provide fault tolerant network
10 filesystem (NFS) data serving.

2. Description of the Related Art:

 With the increasing reliance on computers
 throughout society, increasing emphasis is placed upon
15 providing computer systems with insured availability.
 In general, such systems are referred to as fault
 tolerant computer systems. There are a number of quite
 disparate system implementations of fault tolerance,
 largely due to how critical operations are identified.
20 What constitutes a critical operation is, in turn,
 dependant on the particular type of tasks required of
 such computer systems. Consequently, the particular
 construction of fault tolerancing features implemented
 in any given computer system may and, generally does
25 take significantly different architectural forms.

 Several common fault tolerancing considerations
 exist with regard to most if not all of the disparately
 architected computer systems. These considerations

-2-

5 include minimizing performance loss due to the addition
of fault tolerance capabilities, reducing costs and
maximizing the utilization of hardware and software in
a fault tolerant system, minimizing administration of
the fault tolerant features, limiting the impact of the
10 implementation of fault tolerant features on the end
users of the services provided by the computer system
and, finally, reducing the practical significance of
architectural constraints on the expendability or
modular enhancement capabilities of the computer
15 system.

One form of fault tolerance in computer systems is
known as intra-server fault tolerance. This form of
fault tolerance applies to a unified computer system
utilizing multiple central processor units (CPUs)
20 operating internally in a mutually redundant fashion.
A dedicated error detection control subsystem is
provided typically at the CPU base level to identify
any operational discrepancies in the programs executed
synchronously by the redundant CPUs. A malfunctioning
25 CPU can thus be identified and functionally eliminated
from further participation in providing computer system
services. The remaining CPU or CPUs may then continue
to provide all requested services, pending
identification and correction of the faulty CPU.

30 Conventional intra-server fault tolerant computer
systems, though providing a high degree of reliability,
inherently embody two significant disadvantages. These
disadvantages include a requirement of a highly
constrained or specialized architectural design and a
35 high cost due to substantial duplicated hardware.
Offsetting these disadvantages is the very high degree
of reliability obtained by insuring correct operation

-3-

5 down to an essentially machine cycle level of operation.

Another known form of fault tolerance is obtained through the close coupling of computer systems at the data subsystem level. A typical architecture employing
10 such an inter-server approach utilizes redundant pair of computer systems cross coupled through dual ported disk subsystems. Each system is thus permitted to perform as a master initiator of disk data transfer requests. Should either of the computer systems fail,
15 to the exclusion of the corresponding disk subsystem, the remaining computer system can have complete access to the data of the failed system.

The disadvantages of this form of fault tolerance is that there is little protection against the
20 corruption of data written by a failing system and exposure to a failure occurring in either of the disk subsystems. There is also the significant additional cost of providing and maintaining dual-ported disk subsystems on all of the fault tolerance protected
25 computer systems. An architectural limitation is also present in that dual-porting only allows paired systems to be mutually protected. Also, the paired systems must be mutually local, thereby requiring single site survival as a prerequisite to fault tolerant operation.
30 Finally, a substantial performance penalty is involved at the point any failover. Since the two computer systems only have port-based access to the disk subsystem of the other computer system, the disk subsystem of the failed host must be first cleaned and
35 then remounted by the remaining host. Where large disk subsystems are in use, this process may take several hours.

-4-

5 Finally, another approach for providing fault
tolerance occurs in networked computer environments.
This approach operates by use of network filesystem
shadowing and may be generally implemented as either a
client initiated or a server-only compatibility.
10 Client initiated shadowing requires that the client
computer system send all network operation requests to
two or more pre-established remote server systems.
This approach to fault tolerance allows each client to
independently insure the identity of the client data in
15 two or more physically separate data storage locations.
The client is, however, burdened with the
responsibility of monitoring and analyzing the
completion of all such duplicated requests. The client
must be able to resolve both conventional network
20 transmission errors as well as the loss of access to
any of the remote network server systems.

A clear disadvantage of client initiated shadowing
is, of course, the requirement for modified operating
system software to be executed on each fault tolerant
25 protected client. This approach also admits of
inconsistent administration of the client shadowed data
storage areas on remote servers by requiring each
client to define and control the areas of fault
tolerance. Client initiated shadowing also shifts the
30 responsibility for responding to shadowing errors, and
their correction, to the user of the client.

The known use of NFS shadowing at the server
system level relies on delayed writes of shadowed data
from a primary to a secondary server system. NFS
35 server level shadowing thus requires only the real-time
logging of all data modifications stored on one server
to be replicated to at least the second server. The

-5-

5 inter-server transfer of such logged data is performed
as a low priority background task so as to have minimal
impact on the normal function and performance of the
primary server system. Even so, the delayed background
10 server system may consume a substantial portion of the
network resources of a primary server. Another problem
with NFS server shadowing is that, at the point of any
failover, the delayed write of logged data to the
surviving system creates an exposure window for the
15 loss of data.

Consequently, in view of the necessity of client
software modification and user administration in client
based NFS shadowing and the performance penalties and
real potential for data loss in server based NFS
20 shadowing, the use of NFS based fault tolerant systems
has not been generally well received in the industry
where data fault tolerance is a requirement.

Summary of the Invention

25 Thus, a general purpose of the present invention
is to provide a robust fault tolerant computer system
and control protocol for network filesystem data
transactions.

This purpose is achieved by the present invention
30 through the use of a network computer system providing
for the fault tolerant storage and retrieval of data
files that includes a client system connected to a data
communication network that may source a first data
transfer request to said data communication network for
35 the transfer or retrieval of data. A first server
system, including first medium for storing data files,
is connected to the data communication network so as to

-6-

5 be responsive to first data transfer requests. A
second server system, including second medium for
storing data files is also connected to said data
communication network to also be responsive to first
10 data transfer requests. A control protocol,
established between the first and second server
systems, coordinates an asymmetric response by the
first and second server systems to a first data
transfer request, such that file data transferred by
the client with the first data transfer request is
15 automatically replicated to the first and second
storing mediums and such that file data transferred to
the client system in response to the first data
transfer is non-replicatively provided to the client
system by either the first or second server system.

20 The present invention thus realizes the advantage
of a high degree of fault tolerance protection for data
manipulated within a network filesystem environment
through the use of a specialized NFS transaction
protocol for insuring a robust, fault tolerance
25 mirroring of data among two or more network file
servers.

Another advantage of the present invention is that
failover between a mutually fault tolerance protected
server systems of the present invention are relatively
30 instantaneous. That is, the failure detection aspect
of the protocol of the present invention can detect and
handle failure events consistent with recovery from
normal NFS error conditions.

A further advantage of the present invention is
35 that no additional hardware and minimal additional
software is required for the implementation of the
present invention. The protocol utilizes the same

-7-

5 network connectivity existent for communication with a
client to establish the fault tolerance data
communication path between two or more fault tolerance
server network systems. The only specific hardware
10 disk data storage on each of the server systems that
functions as a mirror back-up to the filesystem storage
of another server system. A related advantage is that
no change is required either to the system software or
hardware of a client station in order to make use of
15 the present invention. Furthermore, administration of
the present invention is centralized on the server
systems.

Yet another advantage of the present invention is
that the fault tolerant network protocol of the present
20 invention may be implemented on both conventional and
specialized server systems and may be further operative
to establish a fault tolerant pairing of server systems
that are not nearly identical.

A still further advantage of the present invention
25 is that fault tolerant operation is established in a
flexible manner that allows any filesystem or other
data object to be established as either a primary or
back-up fault tolerant protected element.
Consequently, load sharing between multiple file
30 servers may be readily established to minimize the
practical implications of establishing fault tolerance
behavior between the file servers with respect to
specific fault tolerance protected filesystems or data
objects.

35 Yet still another advantage of the present
invention is that each primary file server, succeeding
a failover event or other operational data handling

-8-

5 inconsistency, may readily establish a local record of
all data modifications occurring from the point in time
of the failover event, thereby permitting a rapid data
reconstruction of the back-up file server prior to re-
establishing the fault tolerant pairing of filesystems.

10

Brief Description of the Drawings

These and other advantages and features of the
present invention will become better understood upon
consideration of the following detailed description of
15 the invention when considered in connection of the
accompanying drawings, in which like reference numerals
designate like parts throughout the figures thereof,
and wherein:

Figure 1 is a block diagram of an inter-server
20 system suitable for employing the fault tolerant
network filesystem protocol of the present invention;

Figure 2 is a diagram illustrating the normal
client and inter-server protocol transactions necessary
to implement a normal NFS write operation in a fault
25 tolerant NFS mode in accordance with a preferred
embodiment of the present invention;

Figure 3 provides a state transition diagram
illustrating the sequence of operative states executed
by a fault tolerant pairing of NFS servers in
30 accordance with a preferred embodiment of the present
invention in performance of a normal NFS write
operation;

Figure 4 provides a state transition diagram
illustrating the sequence of states executed in the
event of a back-up NFS server failure to complete a
35 client write request in accordance with a preferred
embodiment of the present invention;

-9-

5 Figure 5 provides a state transition diagram illustrating the state transitions in the event of a primary NFS server failure to perform a client write requests in accordance with a preferred embodiment of the present invention;

10 Figure 6 provides a state transition diagram illustrating the states executed by a fault tolerant pairing of NFS servers in execution of a client read request in accordance with a preferred and an alternate embodiment of the present invention; and

15 Figure 7 provides a state transition diagram illustrating the states executed by a fault tolerant pairing of NFS servers in execution of a client create request in accordance with a preferred embodiment of the present invention.

20

Detailed Description of the Invention

A generalized client-server architecture network based computer system 10 is shown in Figure 1. Network data file servers 12, 14 are commonly connected to, in a typical instance, a local area network (LAN) 16. For purposes of the present invention, each of the file servers 12, 14 includes a disk subsystem at least logically partitioned into a number of mutually discrete filesystems. These filesystems represent separately manageable non-volatile disk data storage spaces. Individual client workstations 18, 20 are also connected directly, as shown, or indirectly to the LAN 16. Each of the workstations 18, 20 and file servers 12, 14, in accordance with a preferred embodiment of the present invention, may use any one of a number of variants of the UNIX™ operation system, which is

-10-

5 generally described in Maurice J. Bach, THE DESIGN OF
THE UNIX OPERATING SYSTEM, published by Prentice-Hall,
Inc., Copyright 1986 by Bell Telephone Laboratories,
Incorporated, which is expressly incorporated by
reference herein, and specifically SunOS version ____
10 as distributed by Sun Microsystems, Inc., 2550 Garcia
Avenue, Mountain View, California 94043. The preferred
hardware implementation of the workstations 18, 20 is
any of the workstation products of Sun Microsystems,
Inc. or Sun NFS network compatible workstation products
15 of other manufacturers. The preferred software
implementation of the file servers 12, 14 uses a
modified version of the SunOS, version ____, operating
system, generally as described in Multiple Facility
Operating System Architecture, Application Serial No.
20 08/225,356, filed April 8, 1994, assigned to assignee
of the present invention and which is expressly
incorporated herein by reference. The preferred
hardware implementation of the servers 12, 14 is a
multiprocessor file server as detailed in Parallel I/O
25 Network File Server Architecture, U.S. Patent No.
5,163,131, issued November 10, 1992; Enhanced VMEBUS
Protocol Utilizing Synchronous Handshaking and Block
Mode Data Transfer, Application Serial No. 08/226,398,
filed April 12, 1994; High Speed, Flexible
30 Source/Destination Data Burst Direct Memory Access
Controller, U.S. Patent No. 5,175,825, issued December
29, 1992; Bus Locking Fifo Multi-Processor
Communication System, Application Serial No.
07/933,962, filed August 24, 1992; and High-Performance
35 Non-Volatile RAM Protected Write Cache Accelerator
System, Application Serial No. 08/152,245, filed
November 29, 1993, all assigned to the assignee of the

-11-

5 present invention and which are all expressly incorporated herein by reference.

10 In the preferred operating environment of the present invention, the fundamental transmission control protocol utilized to control the issuance and reissuance of data packets, or datagrams, onto the LAN 16 is a conventional Ethernet (CSMA/CD) protocol. Each datagram provided onto the LAN 16 is encoded with internet addresses (IPs) designating the logical source and destination of the datagram. Each logical entity 15 present on the LAN 16 can thus potentially establish a network dialog with another entity present on the LAN 16 by specifying the assigned internet address of the destination entity in a datagram. The source internet address of the datagram allows the destination entity 20 to responsively identify the unique source of the datagram. Each logical entity on the LAN 16 also nominally uses the destination internet address as a means for screening out datagrams not intended for that logical entity. The use and implementation of the 25 network protocols preferably used by the present invention is described in Andrew S. Tanenbaum, COMPUTER NETWORKS, Second Edition, published by Prentice hall, 1988, which is expressly incorporated herein by reference.

30 In accordance with the preferred embodiments of the present invention, a fault tolerant protocol is implemented for a specific class of remote procedure calls (RPCs) transferred via the LAN 16 as a series of one or more datagrams. Specifically, the class of RPCs 35 encompassed by the fault tolerant protocol include those known as Network Filesystem (NFS) requests. In general, NFS requests provide for two categories of

-12-

5 operations: inquiries and updates. Inquiry requests include read data, get attributes, look up, read directory, read link, status, and null. Update requests include write data, set attributes, rename, remove directory, remove file, link, create file, make
10 directory, and make symbolic link. These NFS requests are monitored and managed by the fault tolerant protocol of the present invention in a manner that results in the mirroring of all data within predetermined filesystems present on a primary 12 and
15 least one secondary 14 file server. The mirroring of data to both the primary and secondary file servers 12, 14 is performed essentially concurrently in response to any client workstation 18, 20 that issues NFS requests with respect to the mirrored filesystems.

20 In brief, the fault tolerant mirroring protocol of the present invention provides for the secondary server 14 to operate in a proxy mode relative to all NFS requests that are otherwise specifically directed to the primary file server 12 and further identified as
25 specific to mirror filesystems physically present on the primary and secondary servers 12, 14. Thus, by proxy operation, the secondary file server 14 concurrently performs all file creation type NFS operations and NFS data update operations requested by
30 any client workstations 18, 20 and directed to the mirror filesystem present on the primary file server 12. Particularly due to the concurrent performance of all NFS data writes, the present invention does not incur the substantial performance penalty of the prior
35 art where serialized data transfers are required in order to maintain synchronization of the data on secondary file servers.

-13-

5 The preferred approach to enabling proxy operation
is to establish a virtual server system formed as a
logical composite of a primary and one or more
secondary servers, constituting an active group,
relative to a specific set of mutually fault tolerantly
10 protected individual filesystems. The virtual server
is identified by a unique hostname and IP address. A
MAC layer multicast address, commonly subscribed to and
accessible by the primary and secondary servers, is
also assigned to the virtual server. Client
15 workstations access the fault tolerant protected
filesystems of the active group by reference to the IP
address and through the MAC multicast address of the
virtual server. Consequently, NFS requests directed to
the virtual server are commonly received by the primary
20 and secondary servers of the active group.

 The fault tolerant protocol of the present
invention provides for the configuration of the primary
and secondary servers so that multiple active groups
can be formed. Each active group thus appears to be a
25 unique virtual server the serves a set of one or more
filesystems. A primary server in one active group may
perform as a primary or secondary server of another
active group.

 The protocol of the present invention includes a
30 set of protocols used to (1) establish and maintain
coordinated operation of the primary and secondary
servers in fault tolerant operation, (2) coordinate the
transfer of data between client workstations and the
primary and secondary servers, and (3) manage the data
35 resynchronization of fault tolerant protected
filesystems.

-14-

5 An exported filesystem map is used to initially
define and establish coordinated operation of the
primary and secondary servers of an active group. An
exported filesystem map is provided on each of the
primary and secondary servers 12, 14. This map
10 establishes the identity of the local filesystems that
are available, or exported, for network access. The
map also defines the access privileges of other network
entities to the exported filesystems. In the preferred
embodiments of the present invention, this map further
15 provides an identification of the mirror filesystems
that are to be served through a virtual server system
and the host name of the virtual server. Each of the
export maps should be identical with respect to the
mirrored filesystems of each active group. The export
20 map also provides a preferred ordered identification of
the actual server systems that are to participate the
virtual server system for each of the mirror
filesystems.

 The exported filesystem map is consulted by a
25 server system whenever a client requests the mounting
of a filesystem. Consistent with the present
invention, a mount request is issued by a client 26
against a mirrored filesystem exported by a virtual
server. The request is therefore multicast to all of
30 the servers of the relevant active group. The primary
server of the relevant active group is responsible for
generating an NFS filesystem ID that is to be returned
to the client 26 for use in future NFS requests. The
primary server 12 is also responsible for initiating a
35 directed network transaction with each of the secondary
servers 14 of the active group to provide the secondary
servers 14 with the mounted filesystem name and

-15-

5 generated NFS filesystem ID. Consequently, each of the servers of the active group maintain a common reference filesystem ID for each exported mirror filesystem.

10 The fault tolerant data protocols of the present invention will be further described in relation to Figure 2. For purposes of description only, the network entities are assumed to have simple, non-subnetted Class "C" IP addresses. Also assumed is that the network entities are capable of mutually transmitting and receiving datagrams using a common MAC
15 multicast address.

A client workstation 26, identified by an internet address of .33, may issue an NFS request to write data to a client NFS mounted filesystem physically associated with a primary file server 12 having a
20 designated internet address (IP) of .121 and a MAC unicast address of :37. Since the NFS write request is to be effectively broadcast on the LAN 16, a secondary file server 14, designated with an internet address of .145 and a MAC unicast address of :64, simultaneously
25 receives the NFS write request 30.

On both the primary files server 12 and secondary file server 14, the datagram representing the NFS write request is processed by a substantially conventional TCP/IP stack. In relevant part, this network stack
30 includes a physical layer, a data link layer, a network layer, a transport layer, a session layer and an application layer.

The physical layer on each primary and secondary server 12, 14 corresponds to the network hardware
35 interface. The data link layer provides the lowest level of datagram integrity checking and flow control. The network layer provides IP routing services

-16-

5 including IP destination filtering. In the preferred
embodiment of the present invention, the network layer
is implemented as a substantially conventional IP
protocol layer, though specifically capable of
10 accepting proxy IP destined datagrams. Proxy operation
is enabled by registering a MAC multicast address, such
as :99, with the network layer. For system management
convenience, a virtual server system internet address
.116 and a unique hostname are correlated to the MAC
15 multicast address of the virtual server by
initialization entries made in the address resolution
protocol (ARP) tables on both the primary and secondary
servers 12, 14. Consequently, datagrams transferred
with reference to the internet address of the virtual
20 server will be automatically multicast on the LAN 16
and both the primary and secondary servers 12, 14 will
receive and accept such datagrams through the network
layers of their respective TCP/IP stacks.

The transport layer provides connection based and
connectionless transport services. The former is
25 preferably implemented through the use of the Transport
Control Protocol (TCP) and the latter through the User
Datagram Protocol (UDP). Each NFS request is, in the
preferred embodiment of the present invention,
implemented as a remote procedure call (RPC)
30 encapsulated in a UDP datagram. The session layer
provides for the establishment of entity sessions based
on an underlying connectionless protocol, including
specifically UDP packets that contain NFS type RPCs.
In the preferred embodiments, the RPC protocol is
35 itself implemented in the session layer.

Finally, the application layer provides for well-
known file services, such as file transfer and remote

-17-

5 file access. An NFS server layer is the preferred
embodiment of the application layer used by the present
invention. Each read, write or other NFS request is
managed through the NFS server under the control of
generally respective network control processes
10 (conventionally *nfsd* processes).

Now considering the processing of datagrams, those
not having a destination IP address of .121 or .116, as
received by the primary server 12, or .145 or .116, as
received by the secondary server 14, are filtered out
15 or blocked in the network layer of the TCP/IP stack.
Particular to the present invention, for those NFS
requests that are destined for IP address .116, the
datagram is routed through the UDP portion of the
transport layer, the RPC portion of the session layer
20 and to the NFS server. If the NFS request is a
permitted operation, as discussed below, the file
request may be processed through the virtual filesystem
switch of a conventional Unix operating system kernel
or directly, as in the preferred embodiment, to the
25 UNIX filesystem (UFS) portion of the operating system.
The UFS 32 is responsible for processing the NFS
originated request whereby filesystem data may be read
or written to disk. Conventionally, the results of
such disk read and write operations are waited on by
30 the UFS 32 before returning any appropriate data and
acknowledging completion of the operation ultimately to
the client workstation 26.

There are several determinations that are made by
the NFS server prior to permitting an NFS request to be
35 processed by the UFS 32. An initial determination must
be made as to whether the request is directed to an
exported filesystem of the virtual server that is

-18-

5 mounted and available. Another determination is whether this actual server is the active primary server or a secondary server for the specific NFS referenced mirror filesystem. These determinations are further made in conjunction with the conventional determination
 10 of whether the request is directed to an exported filesystem for which the requesting client workstation has proper access privileges.

A determination is also made by the NFS server as to whether the specific NFS request is a duplicate of
 15 a prior received and potentially prior processed request. Thus, for each received NFS request, an entry is made in a Duplicate Request Cache (DRC) structure maintained in-memory by the NFS server. This structure, as modified for use in conjunction with the
 20 present invention, is detailed in Table I below.

Table I
Duplicate Request Cache

```

25 struct dupreq {
        u_long      dr_xid;      /* transaction ID */
        u_long      dr_proc;     /* procedure called */
        u_long      dr_vers;     /* version number called */
30  u_long          dr_prog;      /* program number called */ char
        dr_inprogress; /* 1 if request is in progress */
        char        dr_status;   /* status of original reply */
        u_short     dr_port;     /* UDP port of sender */
        struct in_addr dr_hostaddr; /* IP address of sender */
35  struct timeval  dr_timestamp; /* time stamp */
        struct duprply *dr_reply; /* reply for non-idempotent req */
        struct dupreq *dr_next;  /* LRU cache chain */
        struct dupreq *dr_chain; /* hash chain */
40  #ifdef FTNFS
        char        dr_ack;      /* bit mask of backup acks rec'd */

```

-19-

```
5      u_long      dr_inode;      /* new file inode and generation */
      u_long      dr_generation; /* as provided by the primary */
#endif /* FTNFS */
};
```

10

The DRC structure is used to successively record all NFS requests that are accepted for execution by the NFS server. The entries are cached indefinitely, though subject to a least recently used (LRU) entry replacement algorithm. Thus, before an NFS request is accepted and passed to the UFS 30 for execution, the DRC structure is searched to identify any entry that represents a prior duplicate of the present NFS request. This operation of the DRC structure allows for the practical, imperfect nature of the LAN connection between clients and servers wherein datagrams can be lost, corrupted, duplicated, routed out of order, and delayed. Of particular significance is that when NFS requests or their responsive completion datagrams are lost, corrupted, or simply delayed in their delivery, a client workstation may reissue the original NFS request. If the NFS server encounters such a duplicate request, the new request is generally discarded. If the requested operation is currently in progress, the operation continues to its normal conclusion. If the request operation has already completed and a completion datagram sent to the client workstation, though apparently not received, the completion datagram is resent. In general, a repeated completion datagram will ultimately be received by the workstation 26.

In accordance with a preferred embodiment of the present invention, the NFS server code, including

-20-

5 daemon process executed code generally corresponding to
the function of the conventional *nfsds* on each
participating server, is modified to implement the
fault tolerant data protocols. As part of the NFS code
modification, the DRC structure is also modified to
10 include the following new data fields for each DRC
entry. A *dr_ack* field is used as a character sized
bit-map mask of the potentially multiple secondary
servers that have completed this request. Each bit
corresponds to one of up to eight secondary servers
15 mapped in the ordered sequence of such servers as
established in the export map statically initialized
from the */etc/exports* file, as further described below.
The *dr_inode* field is specific to create file, make
symbolic link, and make directory NFS requests. This
20 field is used to store the inode number that is to be
used in completing the NFS request operation; this
field is only used on secondary servers. Finally, the
dr_generation field is again specific to create file,
make symbolic link, and make directory NFS requests.
25 The generation number is preferably a timestamp of the
time the request referenced inode was originally
allocated, typically on a primary server. Since inodes
are reused in the course of cyclical file creation and
deletion, the generation field is used to verify
30 whether a file referred to by an inode number is truly
intended for the current instance of the inode.

The fault tolerant data protocol of an exemplary
NFS write operation is shown in Figure 2. A secondary
server 14 operates as a proxy for the primary 12, at
least with respect to NFS requests directed against a
35 mirrored filesystem present physically on both of the
servers 12, 14. In the preferred embodiment of the

-21-

5 present invention, the file storage space available in
the mirror filesystem of the secondary server 14 is at
least as large as the corresponding mirror filesystem
on the primary server 12. Likewise the number of
10 inodes available in the secondary server mirror
filesystem must be at least as many as allocated for
the primary server mirror filesystem.

In order to establish proxy operation, the MAC
multicast address is programmed into the network layer
of the TCP/IP stack at least upon proxy initialization
15 to pass datagrams having IP addresses that match the IP
address of the virtual server. Thus, in the case of a
NFS write data request 30 directed to a filesystem,
designated `/usr/home1`, physically managed by the server
12 and mirrored by a `/usr/home1` filesystem provided on
20 the secondary server 14, the datagram containing the
NFS write request and accompanying write data is
delivered simultaneously by multicast to both servers
12, 14. The NFS server of server 12 recognizes the
request 30 as directed to a virtual server exported
25 filesystem for which the server 12 (hostname `SvrAB-1`)
is designated as the primary mirror server and that a
secondary mirror filesystem is present specifically on
the secondary server 14 (hostname `SvrAB-2`).

30 Table II
/etc/exports

Syntax:
filesystem_name keyword = ordered list of primary
35 # and secondary servers

-- single secondary server
/usr/home1 -ftnfs=SvrAB-1:SvrAB-2

40 # -- two secondary servers using High Consistency Read protocol
/usr/home2 -ftnfs=SvrAB-1:SvrAB-2:SvrAB-3,HighRead

-22-

```

5  # -- alternately ordered servers to load share
   /usr/home3          -ftnfs=SvrAB-2:SvrAB-1:SvrAB-3

   # -- reversed primary read and write servers
   /usr/home4          -ftnfsr=SvrAB-1:SvrAB-2,ftnfsw=SvrAB-2,SvrAB-1
10

```

This information is dynamically obtained by the server
 12 from the in-memory export map initially constructed
 from the `/etc/exports` data file statically stored by
 15 the server 12. The preferred format of this data file,
 as stored in an otherwise conventional `/etc/exports`
 data file, is generally provided in Table II.

In similar manner, the secondary server 14 also
 recognizes that the request is directed to a file
 20 system exported by the virtual server. By the ordering
 of the actual active server hostnames for the
`/usr/home1` filesystem, identically determined from the
 secondary server's own `/etc/exports` file, the server 14
 self-determines to be the secondary mirror server with
 25 respect to this file system.

In normal operation then, the request 30 is
 processed through to the NFS server layers of both the
 primary and secondary servers 12, 14. Both create
 corresponding DRC entries. The NFS server of the
 30 primary server 12 ultimately passes the write data
 request to the UFS 32. The write data operation 40 is
 then executed 40 by the UFS 32. In parallel, the write
 data request is ultimately passed and executed 42 by
 the UFS 34 of the secondary file server 14. On
 35 completion of the write operation, the UFS 32 write
 process 40 sleeps 44 pending an update of the DRC entry
 corresponding to the write request 30. That is,
 specific for mirrored filesystems in accordance with

-23-

5 the present invention, the write data process 40 does
not immediately pass through to the generation of a
completion datagram to the client on completion of the
write operation. Rather, the controlling *nfsd* process
40 goes to sleep on a wait system call pending an
10 update to the *dr_ack* field of the corresponding DRC
entry. This field is updated in response to the
receipt of an acknowledge datagram 46 from the
secondary server 14 issued as a consequence of
completion of the write operation 42.

15 On update of the last acknowledge field bit for
the known set of secondary servers 14, the sleeping
write process 40 is awakened 48 and execution continues
with the generation 50 of a completion datagram 52 that
is then returned to the client workstation 26.

20 Although the primary server 12 may sleep on
completion of the write operation, and thereby
introduce a delay in the generation and transfer of a
completion datagram to the client workstation 36, this
delay is small relative to the transfer time of large
25 block NFS write datagrams and the absolute period of
time required to write the data to the physical
filesystem. The controlling factor in the length of
the delay imposed by the present invention is then
largely the relative difference in time required by the
30 file servers 12, 14 to each complete the write data
operation 40, 42. Where the secondary server 14 is
relatively less busy, resulting in a more rapid
execution of the write operation 42, the acknowledge
field of the corresponding DRC entry in the structure
35 36 may be updated prior to completion of the write
operation 40. Consequently, the added sleep delay is
zero. Where the primary server 12 is relatively less

-24-

5 burdened, the sleep delay is substantially equal to the
difference in time required by the secondary server 14
to complete the write operation 42. In all normal
events then, the additional delay imposed by operation
of the present invention is generally only the
10 difference in the actual write operation completion
time of the servers 12, 14. Thus, the servers 12, 14
can be seen to be operating in a parallel, asynchronous
or a loosely concurrent relationship that exhibits
minimal re-synchronization delay in order to obtain and
15 maintain identity between the data stored by
mirrored filesystems on the servers 12, 14.

Referring now to Figure 3, a more detailed control
and data flow 60 describing a client write request
operation is shown. The flow 60 illustrates the states
involved in a successful completion of the NFS write
20 request. The servers 12, 14 each receive and qualify
the request 62, 64. Corresponding entries are made in
the DRC table structures. The servers then
asynchronously perform the write data requests 66, 68.
25 The primary server 12, on completion of the write,
initially determines whether an acknowledgment message
corresponding to this write request has been received
from the secondary server 14. Where an acknowledge has
not yet been received, the controlling *nfsd* server
30 process sleeps 70 waiting for the expected eventual
receipt of an acknowledgment datagram.

When the secondary server 14 completes execution
of the write request, an acknowledgment datagram is
prepared and sent 72 to the primary server 12. The
35 acknowledge datagram includes the client specified
transaction ID and client IP address (*dr_xid*,
dr_hostaddr) as specified in the originating NFS write

-25-

5 request and recorded in the DRC entry on both the
primary and secondary servers 12, 14. By using the
information contained in the acknowledge datagram, the
primary server 12 updates the appropriate *dr_ack* bit
field in the DRC structure 36. The specific DRC entry
10 is found by searching the DRC structure for a DRC entry
with the same transaction ID and client IP address
(*dr_xid*, *dr_hostaddr*). The specific *dr_ack* bit to
update is determinable from the IP source address of
the acknowledge datagram and by reference to the
15 ordered list of secondary servers in the in-memory copy
of the exports map. When all expected acknowledges
have been received, the specific sleeping *nfsd* process
is found by correlation to the position of the DRC
entry in the DRC structure. The *nfsd* process is then
20 awakened. The NFS write request process on the primary
server 12 then continues on to the preparation of a
conventional NFS completion datagram that is sent to
the client 76.

Operation of the present invention in a secondary
25 server failure scenario is generally illustrated in
Figure 4. Again, a client write request is issued by
the client 26. The request is received and performed
62, 66 by the primary server 12. As before, the
corresponding *nfsd* process on the primary server 12
30 then sleeps relative to the specific write request
awaiting an acknowledgment datagram from the secondary
server 14.

The primary server 12 may fail to receive an
acknowledge datagram from the secondary server 14 for
35 a number of reasons. The secondary server 14, in a
first instance, may have failed to properly receive the
client write request. Alternately, the secondary

-26-

5 server 14, in performing the request, may fail for some
reason to complete the request. In either case, no
acknowledgment datagram is issued by the secondary
server 14. Another possibility is that the client
10 write request was properly received and performed by
the secondary server 14, but the issued acknowledgment
datagram was not properly received by the primary
server 12.

In each of these events, the primary server 12 is
left sleeping on the DRC entry for an acknowledgment
15 datagram that is not received. However, in accordance
with the present invention, a sleep timer is set by the
primary server 12 in putting the *nfstd* process to sleep
on DRC entry. The *nfstd* process awakes 86 on timeout of
the sleep timer in the absence of any received
20 acknowledge datagram. Alternately, the sleep timer is
effectively expired upon the aging of the DRC entry
through operation of the DRC-LRU algorithm. In either
event, the primary server 12 then transitions to a
backup failure recovery mode 88.

25 Where the backup failure occurs in a circumstance
where the mirrored filesystems of the active group
continue to be properly available, and the integrity of
the virtual server is intact but for the failure to
receive the acknowledgment datagram, a partial
30 resynchronization of the mirrored file systems is
possible. The availability of mirrored filesystems is
established by use of a heart-beat protocol, preferably
performed on a per mirrored filesystem basis by all
active group servers on the LAN 16, to continually
35 broadcast evidence of the continuing availability of
the corresponding exported filesystems on the LAN 16.
Preferably, this heart-beat protocol is implemented

-27-

5 through the issuance of a custom UDP datagram multicast
to the file servers of the active group. Where such
heart-beat datagrams are still being exchanged between
at least the active group servers 12, 14 of a given
10 mirrored filesystem, thereby indicating that the mirror
filesystem on the secondary server 14 is available even
though a sleep event for an acknowledge packet has
timed out on the primary server 12, the primary server
12 may intentionally withhold issuing any completion
15 datagram to the client 26. Subject to the conventional
operation of the NFS protocol, the client 26 will
ultimately time-out waiting for the completion datagram
and reissue the NFS write request.

Alternately, the primary server 12, prior to
issuing a completion datagram for the NFS write
20 request, may itself initiate a conventional NFS write
operation directly to the secondary server 14 to force
completion of the client requested NFS write operation.
In this latter case, the NFS writes to the affected
virtual server filesystem location must generally be
25 ordered and queued pending completion of the retry as
generally provided for by conventional locking
protocols. In both instances, a limited number of
either client or primary server 12 retries are
monitored by at least the primary server 12 before a
30 failover event is declared. Of course, should a retry
be successful, the number and nature of the retries are
preferably logged by the servers 12, 14 and a
completion datagram is sent to the client 26.

If a failover event is declared by the primary
35 server 12, the in-memory export map maintained by the
primary server 12 for the virtual server is updated to
effectively delete the secondary server references for

-28-

5 the affected filesystem. Where other secondary servers
exist for the affected filesystem, the primary server
12 maintains operation of the fault tolerant protocol
of the present invention with respect to those
remaining secondary servers. If all secondary servers
10 have failed, the primary server continues operation
relative to the affected filesystem generally in a
conventionally NFS filesystem server mode.

The primary server 14 will declare a failover
event on a per filesystem basis immediately on failure
15 to receive a heart-beat datagram for a given mirrored
filesystem within a user programmable threshold time
period. The absence of these datagrams indicates that
corresponding filesystem of the secondary server 14 is
unavailable or that there has been a hard network
20 communications failure between the primary and
secondary servers 12, 14.

Once a failover event is declared, the primary
server 12 ultimately issues a completion datagram to
the client workstation 26 indicating a proper
25 completion of the pending NFS write request. The
primary server 12 also preferably logs the failover
event and issues appropriate status messages to the
administrator's console.

Referring now to Figure 5, the failover scenario
30 associated with a primary server failure is shown. A
client write request is properly received by the
secondary server 14 and performed 64, 68. An
acknowledgment datagram is generated 72 and issued to
the primary server 12. However, either due to an
35 initial failure to receive the write request 90 or a
failure in execution of the request 92, the client
write request was not performed. In either case, a

-29-

5 corresponding DRC entry does not exist; the DRC entry
never existed or was discarded by the primary server 12
on encountering a write execution error. Consequently,
the acknowledgment datagram from the secondary server
14 is received by the primary server 12 as an
10 acknowledgment for an unknown request. The primary
server 12 then transitions to a failure handling state
96.

The primary server 12 preferably sends an alert
datagram to the secondary server 14. The alert
15 datagram contains sufficient information to identify
the unknown acknowledgment datagram as received by the
primary server; specifically the transaction ID and
client IP address provided in the unknown
acknowledgement datagram. The secondary server 14
20 could, in symmetry to the primary server failover
scenario, initiate an NFS write request to the primary
server 12 to force success of the original client write
request. However, a policy is preferably implemented
to the effect that the data on the active primary
25 server for a given filesystem is to be considered at
all times to be correct, even in the event of a write
failure. Preferably then, the secondary server 14
generally only counts and logs the receipt of such
alert datagrams. The primary server 12 also counts and
30 logs but otherwise generally ignores the unknown
acknowledge datagram. Both the primary and secondary
servers 12, 14 rely on the client 26 to repeat any
unacknowledged NFS write request.

However, should either the primary or secondary
35 server 12, 14 recognize that a user programmable
threshold number of unknown acknowledge or
corresponding alert datagrams have been received, a

-30-

5 failover event is declared to exclude the primary server 12 from the active group mirroring the affected filesystem.

Another primary server failure scenario occurs on a partial failure of the primary server 12. In this instance, the secondary server 14 will recognize that a filesystem of the primary server 12 is unavailable by a failure recorded by the heart-beat protocol. In this event, the secondary server 14, assuming that the server 14 is the first listed secondary server, updates its in-memory export map to exclude active group participation by the primary server 12 relative to the affected filesystem and to establish the secondary server 14 as the new active primary server for the affected filesystem. Any other secondary servers similarly update their in-memory export maps to recognize secondary server 14 as the new primary server. In all events, no acknowledgment datagram is sent to the primary server 12 by the secondary server 14 at state 72. Rather, a conventional NFS completion datagram is sent directly to the client 26 by the secondary server 14.

Where a secondary server 14 loses all communication with an active primary server 12, the secondary server 14 should not assume that the primary server 12 has failed. Rather, the likely failure point is the LAN 16. In that case, the possibility of multiple active primary servers 12 serving the same file system must be avoided. Therefore, a secondary server 14 should not be promoted to an active primary state where all communication with an active primary server 12 is lost.

-31-

5 The present invention provides three methods of
satisfying a client read request, with each method
providing a greater degree of synchronization between
the primary and secondary servers 12, 14. The first
two methods of responding to a client read request are
10 generally shown in Figure 6. The first and simplest
method of performing a client read request is the
asynchronous read. For an asynchronous read, the
primary server 12 receives the client read request 100,
performs the request 102, and then immediately
15 acknowledges completion of the read request to the
client workstation 26.

 The client read request is also received 106 in
parallel by the secondary server 14. The read request
itself is ignored, though a DRC entry is created in the
20 DRC structure 38 on the secondary server 14. The last
access time field associated with the referenced inode
of the read request is also updated on the
corresponding filesystem to essentially the time that
the read request was received and processed by the
25 secondary server 14. This updated access time may
differ from the access time recorded for the
corresponding inode on the primary server 12. However,
the significance of the difference in time is mitigated
by the fact that few programs are sensitive to such a
30 slight difference in the last access timestamp of a
data file. Further, the access times recorded by the
secondary server will be biased in a generally
consistent manner relative to the access times recorded
on the primary server 12.

35 Where assurance that access times are properly
recorded by the secondary server 14 or where a higher
level of robustness in performing the read request is

-32-

5 desired, an acknowledged read operation may be
implemented. An acknowledged read differs from the
asynchronous read in that the server process on the
primary server 12 will sleep 110 on the primary DRC
entry corresponding to the client read request rather
10 than immediately issuing a completion datagram 104 to
the client 26. The operation of the secondary server
14 similarly differs in that an acknowledgment datagram
is generated 112 and issued to the primary server 12
following a successful update of the inode access time
15 on the secondary server 14. The corresponding server
process on the primary server 12 will then wake 114 in
response to receipt of the acknowledgment datagram and
proceed to generate and issue 104 a completion datagram
to the client workstation 26 to provide the read data
20 and signal completion of the read request.

Finally, a fully serialized method of responding
to a client read request is shown in Figure 7. The
states executed by the primary and secondary servers
12, 14 are essentially the same for performing a
25 serialized read as used in responding to a client
create request, as will be discussed subsequently.

The fully serialized client read operation
establishes essential identity in the inode information
maintained on both the primary and secondary servers
30 12, 14 for the request referenced read data file. This
is accomplished by establishing multiple interactions
between the primary and secondary servers 12, 14 to
transfer last access time inode information to the
secondary server 14.

35 In operation, the primary server 12 receives 120
a client read request and performs 122 the requested
operation. Concurrently and so far asynchronously, the

-33-

5 secondary server 14 also receives the request 124. As
with the prior methods of responding to read requests,
the secondary server 14 effectively ignores the read
request, though a DRC entry is prepared and entered in
the DRC structure 38 on the secondary server 14. The
10 secondary server 14 read server process then sleeps 126
on the DRC entry.

The primary server 12, on completion of the read
request 122, prepares a send datagram 128 containing at
least the last access timestamp of the inode read in
15 response to the client read request, the transaction
ID, inode number and the generation of the inode. This
datagram is then sent to the secondary server 14 to
wake 132 the sleeping server read process on the
secondary server 14. The secondary server 14 then
20 updates 134 the corresponding inode with the last
access timestamp provided. The secondary server 14
then returns 136 an acknowledgment datagram to the
primary server 12.

In response, the appropriate read server process
25 on the primary server, having slept 130 on the DRC
entry corresponding to the read request, is awakened
138 in response to the receipt of the acknowledge
datagram. A completion datagram is then prepared 140
and forwarded to the client workstation 26.

30 As can be seen, the inode date information present
on both the primary and secondary servers 12, 14 is
maintained identically by use of the serialized read
method. Identity of inode date information in this
manner, however, requires serialized updating of the
35 inodes on the primary and secondary servers 12, 14. A
cumulatively significant though atomically small delay

- 34 -

5 may be incurred through the use of the serialized read method.

10 In accordance with the present invention, the handling of the create NFS requests requires that the primary and secondary servers 12, 14 allocate directly corresponding inodes for files created on the mirrored filesystems. For these operations, the use of a serialized request response method is desirable.

15 The create NFS operations include: file creation, directory creation, and symbolic link creation. In each of these create operations, the essential requirement is the establishment and maintenance of uniquely corresponding inodes in each of the mirror filesystems. This is accomplished in accordance with the present invention by the transmission, from the primary 12 to the secondary server 14 and any other secondary servers, of selected inode related information first generated by the primary server 12 in performing a NFS create request 122. At least the allocated inode number, generation number and file creation timestamp, are sent 128 as part of a send datagram from the primary server 12 to the secondary server 14 and any other secondary servers. This information is then uniquely passed by the NFS client of the secondary server 14 to the UFS 34 for use in actually performing the create request operation 142. Conventionally, the UFS 34 would independently allocate a new inode within the designated filesystem in performing a file creation operation. However, by forcing all inodes in the secondary mirror filesystem to be allocated identically as in the primary mirror filesystem, the primary server 12 can preemptively specify the new inode number to be used in the file

20
25
30
35

-35-

5 creation operation performed by the secondary server
14. Naturally, before using the primary server
specified inode number, the secondary server 14
preferably checks to ensure that the specified inode is
not presently in use. If determined to be in use, a
10 error has occurred and a failover event is declared
against the affected filesystem on the secondary server
14.

Consequently, so long as client create requests
are successfully executed by the secondary server 14,
15 each inode on a mirror filesystem maintained by the
secondary server 14 is created in each corresponding
mirror filesystem with an identical inode number
relative to the primary server 12. Consequently,
subsequent client read and write requests that identify
20 the request against a specific filesystem, inode number
and generation can be commonly performed by both the
primary and secondary servers 12, 14 with the assurance
that the request is performed against mirrored data.

A further and desirable consequence is that, on
25 the occurrence of a failover event, no exceptional
action must be taken with regard to the client
workstations that have mounted a filesystem from the
virtual server. All NFS file handles held by client
workstations remain equally valid as against both the
30 primary and secondary servers 12, 14 of the virtual
server. Indeed, the client workstation 26 is largely
unaffected and unaware of the occurrence of any failure
or failover as between the servers 12, 14.

Another consideration dealt with by the present
35 invention is the recovery of mirrored status between
mirror filesystems on the primary and secondary servers
12, 14 following the correction of the cause of a

-36-

5 failover event. A number of different approaches to
recovery are contemplated by the present invention.
The simplest recovery technique is to simply quiesce
the failover surviving server and copy all data files
10 within the surviving mirror filesystem to the mirror
filesystem of the rejoining server. This copy can be
performed selectively based at least on last
modification timestamps that are subsequent to the
moment of the failover event. The exact time of
occurrence of the failover event is preferably recorded
15 at the time of occurrence by the surviving server
through the `/etc/syslogd` service or an equivalent event
logging service.

The preferred partial resynchronization approach
to recovery is to provide for the logging of all file
20 data modifications, including file creations and
deletions, that are made to the mirror filesystem of
the surviving server from the point of the failover
event to the quiescing of read/write activity in
preparation for recovery. The backup logs may store
25 the accumulated incremental changes to the data files
present on the mirror filesystem. While the mirror
filesystems are otherwise quiesced, this log may simply
be transferred or replayed from the surviving server to
the rejoining server, thereby resynchronizing the data
30 present on the mirrored filesystems. Where write
activity is not quiesced on the surviving server during
the partial resynchronization, a new log of write
information can be accumulated by the surviving server
while writing a prior log to the rejoining server.
35 Successive logs will be smaller until either no writes
occur to the affected filesystem during the transfer of
a log or, once the current log is of a sufficiently

-37-

5 small size, writes to the filesystem on the surviving
fileserver are temporarily suspended or held off until
the final log is written to the rejoining server.

10 Finally, where the failover event is the result of
a catastrophic failure of a server, the mirror
filesystem of the rejoining server may need to be
completely reconstructed using a full resynchronization
of the affected filesystem. In this event, the
filesystem of the surviving file server is quiesced and
an entire filesystem copy performed to transfer an
15 image copy of the surviving filesystem to the rejoining
server. Significantly, this disk copy must be
performed so as to preserve the inode number and
generation designation associated with each of the
files present on the mirror filesystems.

20 Once a data identity has been re-established
between the mirror filesystems of the primary and
secondary servers 12, 14, the filesystem export map
stored in memory by both the primary and secondary
servers 12, 14, and any other secondary servers, can be
25 updated to reflect subsequent operation subject to the
fault tolerant mirroring protocol of the present
invention. The mirror filesystems can then be placed
in an active state available for service with respect
to client requests.

30 Finally, the present invention provides for a
negotiation between the file servers of an active group
to establish and, as necessary, change roles as primary
and secondary servers in the active group. During
ordinary startup operations, the mirror filesystems
35 that combine to form a filesystem of a virtual server
cannot be assured to become available from each of the
servers of the active group at the same time.

-38-

5 Consequently, the new availability of a mirror
filesystem is treated in the same manner as the
filesystem of a fileserver rejoining the active group
following a failover. Thus, the first fileserver of an
active group, preferably with manual confirmation, will
10 generally become the active primary for a given mirror
filesystem, though the fileserver may only be listed as
a secondary server for that filesystem. A partial
resynchronization log is started at least with the
first client workstation write to the filesystem. If,
15 however, the partial resynchronization log has reached
a user programmable size, the logging of write data
will have been terminated. In that case, a full
resynchronization of the mirror filesystems is
required.

20 As a second fileserver, serving the same mirror
filesystem, completes its startup, a partial or full
resynchronization is performed. The mirror filesystem
of the second fileserver is then included in the
operation of the virtual server for that filesystem.
25 However, the rejoining server may be the /etc/exports
listed primary server for the rejoining filesystem.
Preferably then, the active primary fileserver will
preferably switch to its listed secondary role for the
affected filesystem and the rejoining server will
30 assume the role of the active primary.

 A fault tolerant NFS mirroring protocol has thus
been described. The protocol enables substantially
asynchronous mirroring of filesystems on physically
separate file servers necessarily interconnected only
35 by the pre-existing network LAN. All client requests
directed to the mirrored filesystems proceed as
ordinary NFS request operations and, by operation of

-39-

5 the present fault tolerant protocol, result in concurrent operations on the separate file servers to implement the mirroring of the NFS data.

10 Naturally, many modifications and variations of the present invention are possible in light of the above disclosure. These modifications may include the use of redundant, alternate and concurrent LAN or other logical data connections to interconnect the client with the involved filesystem mirroring file servers. The protocol may also be applied to transport services
15 other than the NFS class of UDP services. In particular, the protocol may be extended to cover TCP/IP protocols and other sets of RPC transactions. Finally, it should be recognized that the protocol of the present invention is not limited to inter-
20 operability among the products of a single vendor, but may be implemented independent of the specific hardware and software executed by any particular file server. Accordingly, it is therefore to be understood that, within the scope of the appended claims, the present
25 invention may be practiced otherwise than as specifically described herein.

-40-

5

Claims

1. A computer system providing for the fault tolerant storage and retrieval of data files, said computer system comprising:

10 a) a client system connected to a data communication network, said client system providing a first data transfer request to said data communication network;

15 b) first server system, including first means for storing data files, connected to said data communication network, said first server system being responsive to said first data transfer request;

20 c) second server system, including second means for storing data files, connected to said data communication network, said second server system being responsive to said first data transfer request; and

25 d) control means, coupled between said first and second server systems, for coordinating an asymmetric response by said first and second server systems to said first data transfer request, such that file data transferred by said client with said first data transfer request is replicated to said first and second storing means and such that file data transferred to said client system in response to said
30 first data transfer is non-replicatively provided to said client system by either of said first and second server systems.

35 2. The computer system of Claim 1 wherein data referenced by a predetermined data transfer request is stored by said first and second server systems referencable by a common data identifier.

-41-

5 3. The computer system of Claim 2 wherein said
first and second server systems each include means for
determining whether data referencable by said common
data identifier is transferrable by the other one of
said first and second server systems.

10 4. A server computer system providing for the
fault-tolerant serving of data mutually communicating
over a network with another server computer system and
a client system in response to a network request having
15 a source address and a destination address, said server
computer system comprising:

a) a mass storage device; and

b) a processor couplable to a communications
network, said processor being coupled to said mass
20 storage device and providing for the exchange of data
between said communications network and said mass
storage device in response to a predetermined request
for the transfer of data by a client system, said
processor including

25 means for defining qualifications
applicable to the transfer of predetermined data; and

 means, capable of distinguishing between
a multiplicity of destination addresses, for processing
network requests having one of a plurality of
30 destination addresses, said processing means selecting
between a plurality of response processes based on the
destination address of said predetermined request, at
least one of said response processes providing for the
confirmation of the mutual performance of said
35 predetermined request with another server computer
system.

-42-

5 5. A fileserver system comprising:

 a) a first server system including a first
mass storage device providing a first data storage
area; and

 b) a second server system including a second
10 mass storage device and providing a second data storage
area, said first and second server systems being
coupleable to a network that provides for the
transmission of network requests between a client
system and said first and second server systems, said
15 first and second server systems including means for
processing network requests having a common network
destination address, said first and second server
systems each including means for coordinating the
loosely concurrent processing of each of said network
20 requests.

 6. The fileserver system of Claim 5 wherein said
coordinating means provides for the selection between
a primary and a secondary process of processing a
25 predetermined network request issued commonly against
said first and second data storage areas, said
coordinating means providing for the selection of a
single primary process among said first and second
server systems for said first and second storage areas.

 7. The fileserver system of Claim 6 wherein said
coordinating means provides for the generation of a
common identification datum for said first and second
data storage areas and wherein said single primary
35 process provides for the generation and transmission of
said common identification datum to said client system.

1/5

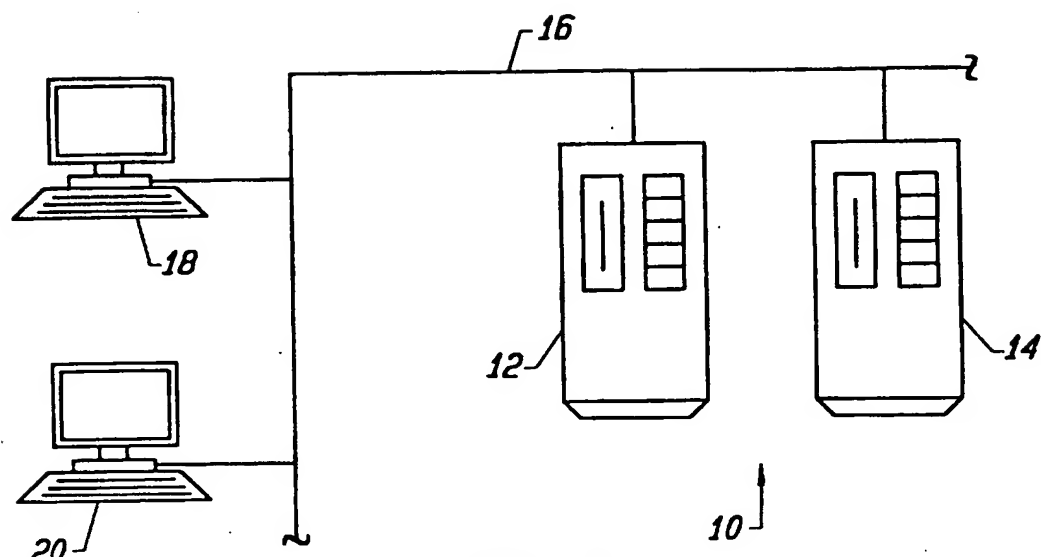


FIG. 1

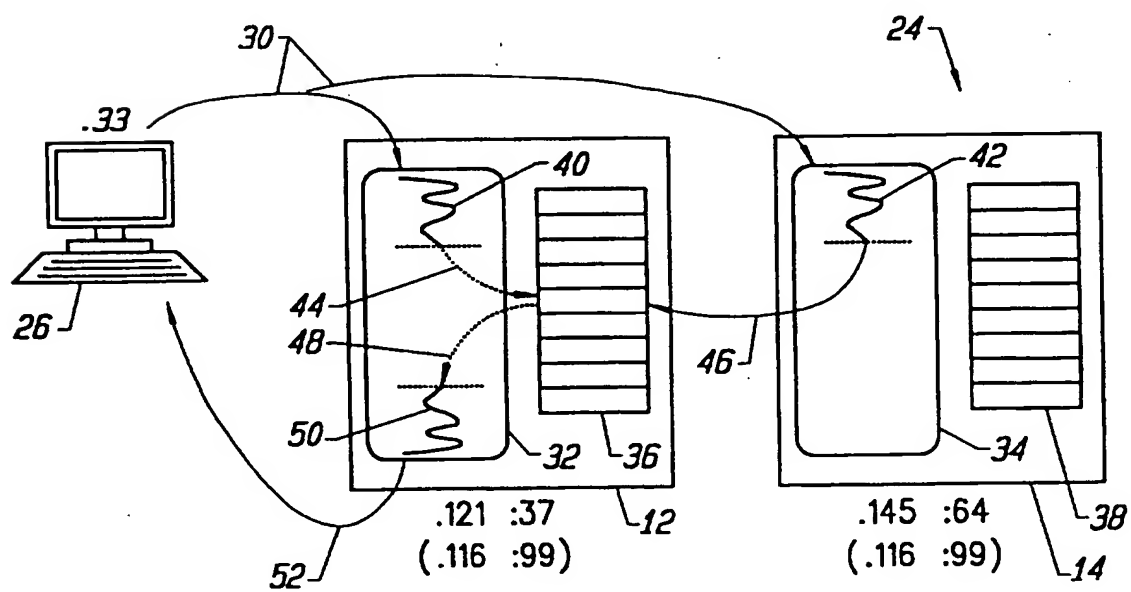


FIG. 2

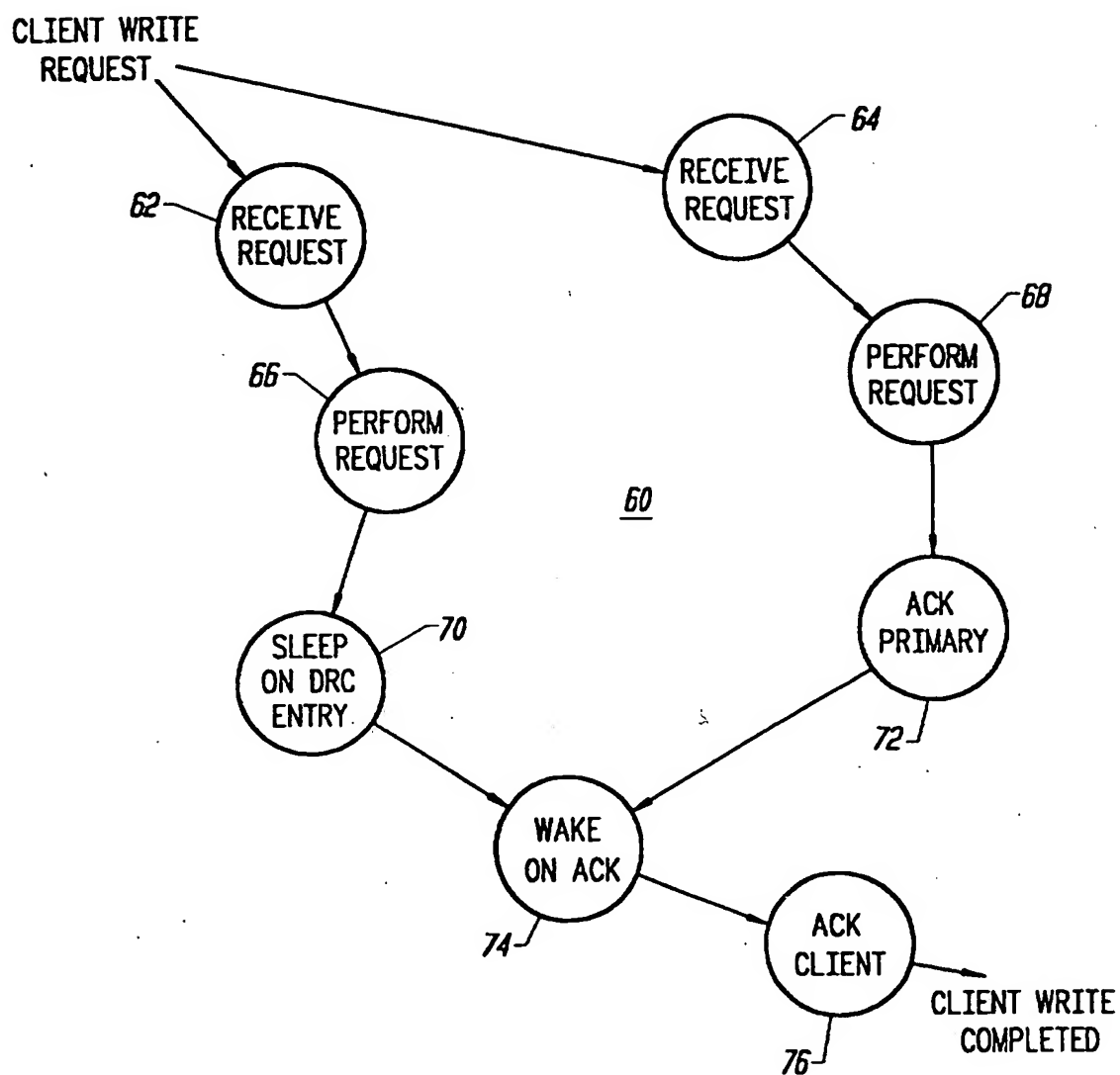
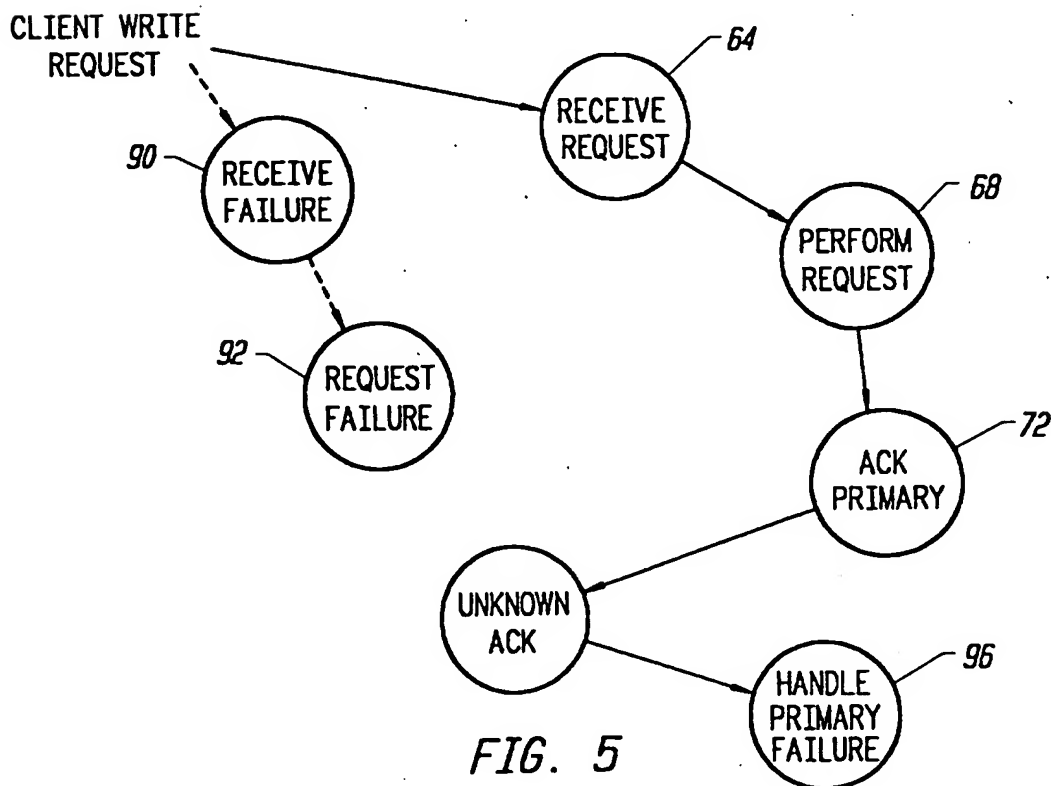
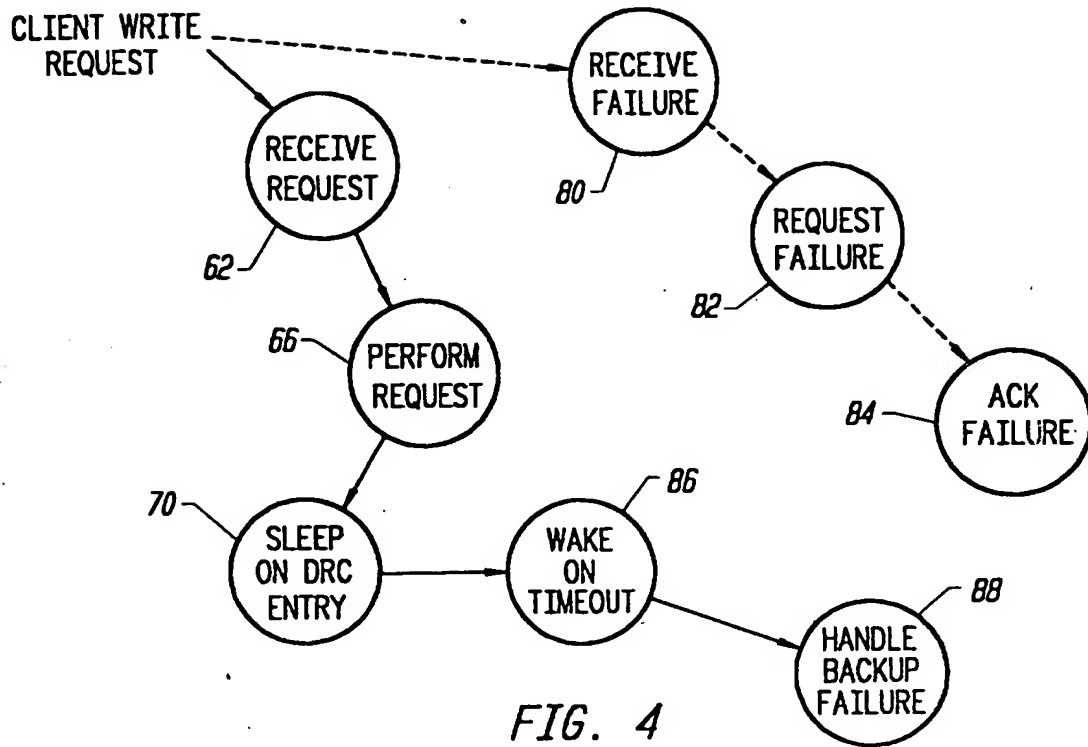


FIG. 3

3/5



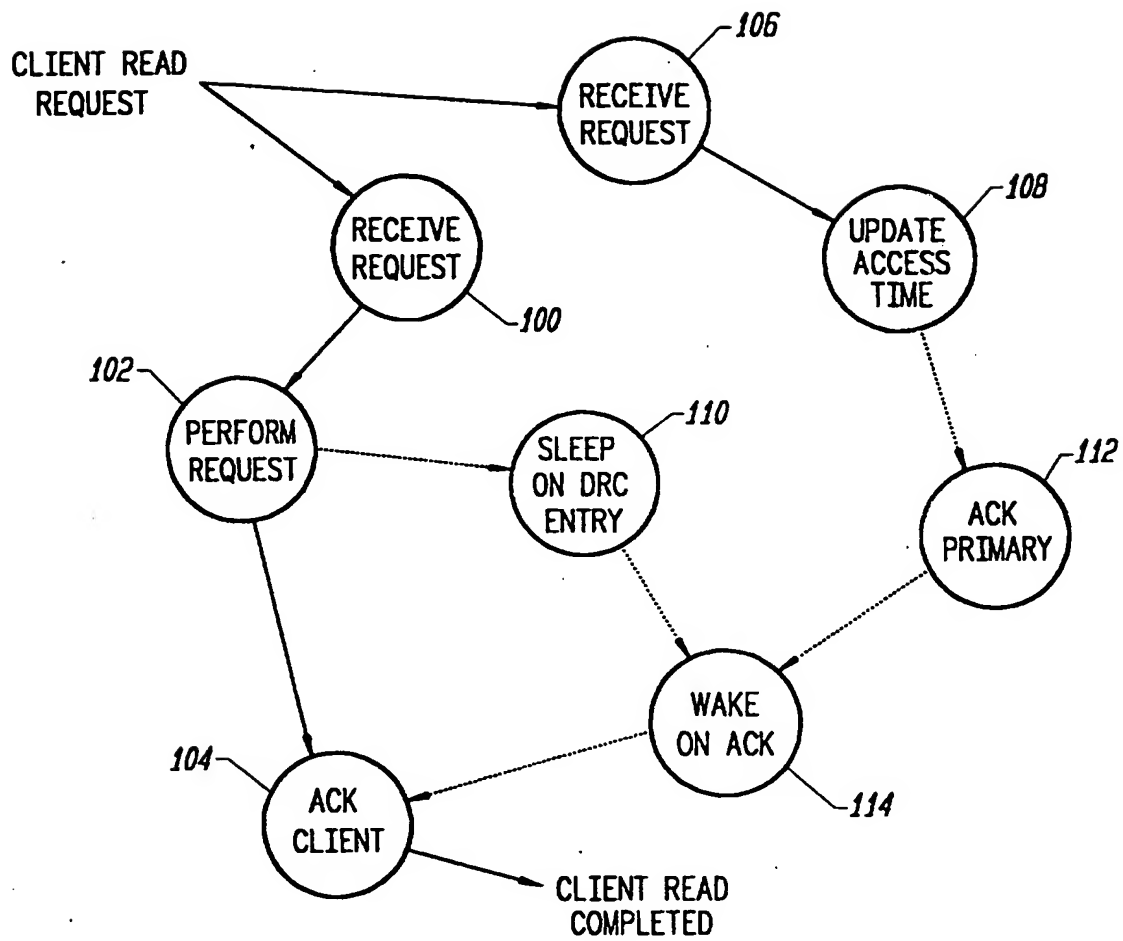


FIG. 6

5/5

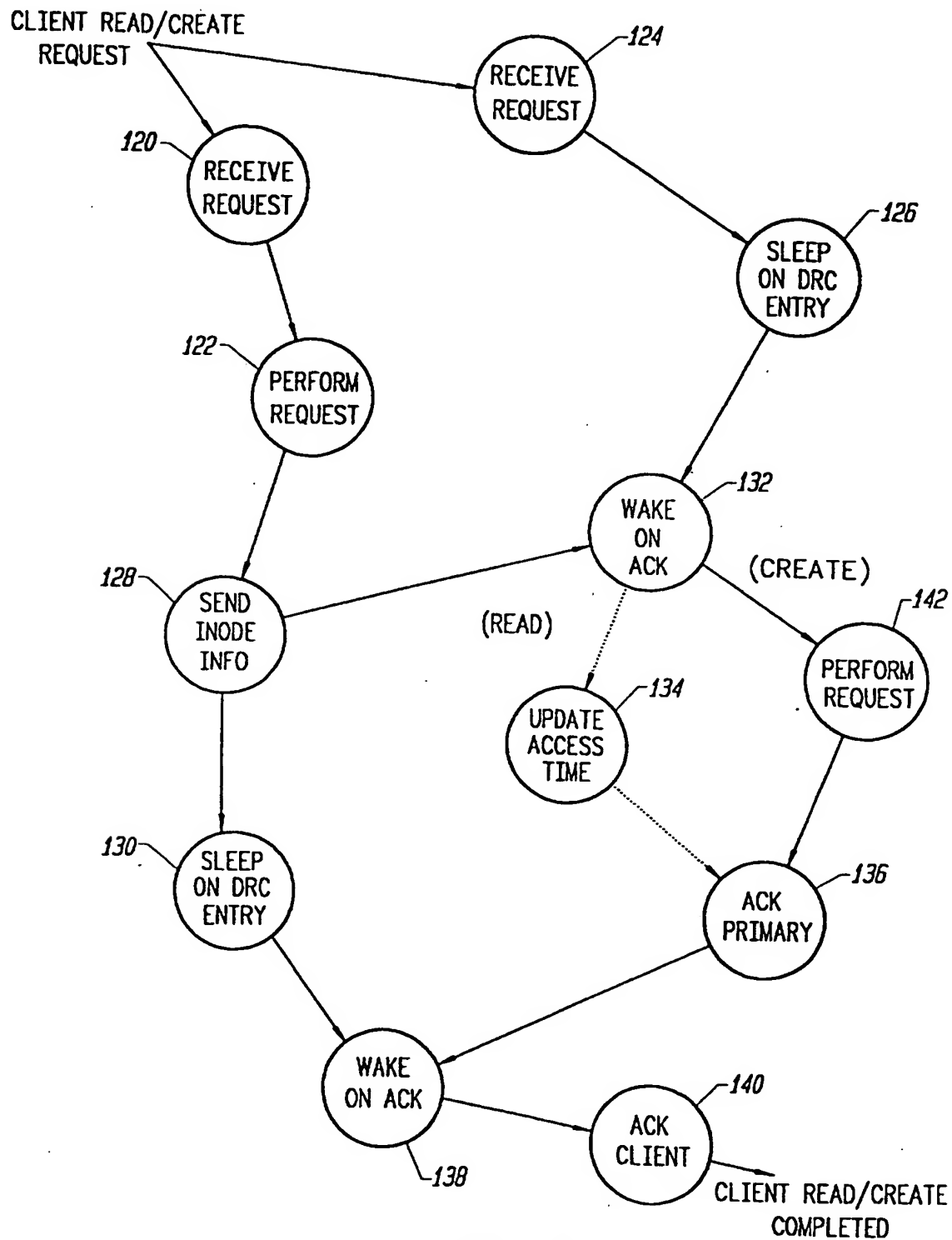


FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/01007**A. CLASSIFICATION OF SUBJECT MATTER**

IPC(6) :G06F 11/00

US CL :395/182.04, 183.18; 371/10.2; 364/268.3, 268.8, 268.9, 269.3

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/182.04, 183.18; 371/10.2; 364/268.3, 268.8, 268.9, 269.3

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS search terms: fault toleran?, network file?, replicate, duplicat?, mirror?, protocol

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US, A, 5,379,418 (Shimazaki et al.) 03 January 1995, col 7, lines 18-53.	1-7
A	US, A, 5,257,369 (Skeen et al.) 26 October 1993	

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:	T	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
A document defining the general state of the art which is not considered to be part of particular relevance	X*	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
E earlier documents published on or after the international filing date	Y*	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
L document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	A*	document member of the same patent family
O document referring to an oral disclosure, use, exhibition or other means		
P document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search

02 APRIL 1996

Date of mailing of the international search report

23 APR 1996

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Authorized officer

Vincent Canney

Facsimile No. (703) 305-3230

Telephone No. (703) 305-9682